1. Importing nessasary libraries

In [21]:

```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import statsmodels.formula.api as smf
import warnings
warnings.filterwarnings('ignore')
```

2. Importing data

```
salary_data = pd.read_csv('Salary_Data.csv')
salary_data
```

In [6]:

```
salary_data.head()
```

Out[6]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

3. Data Understading

In [7]:

```
salary_data.shape
```

Out[7]:

```
(30, 2)
```

In [8]:

```
salary_data.isna().sum()
```

Out[8]:

```
YearsExperience    0
Salary             0
dtype: int64
```

In [10]:

```
salary_data.dtypes
```

Out[10]:

```
YearsExperience    float64
Salary             float64
dtype: object
```

In [11]:

```
salary_data.describe(include='all',)
```

Out[11]:

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

4. Renaming Columns

In [16]:

```
salary_Data = salary_data.rename(columns={'YearsExperience':'experience_data','Salary':'sal
salary_Data
```

Out[16]:

	experience_data	salary_data
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

In [17]:

```
salary_Data
```

Out[17]:

	experience_data	salary_data
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

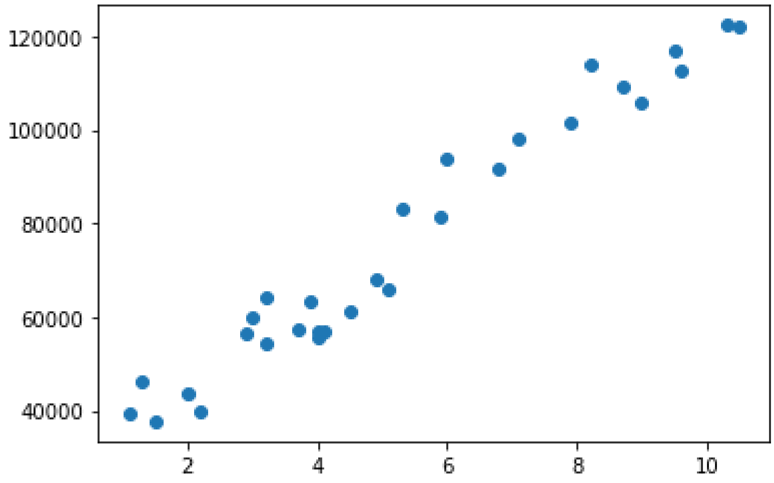
4.[b] Check Assumptions are matching

In [24]:

```
plt.scatter(x = 'experience_data', y = 'salary_data',data=salary_Data)
```

Out[24]:

```
<matplotlib.collections.PathCollection at 0x1b538ad5100>
```



In [25]:

```
# correlation analysis
salary_Data.corr()
```

Out[25]:

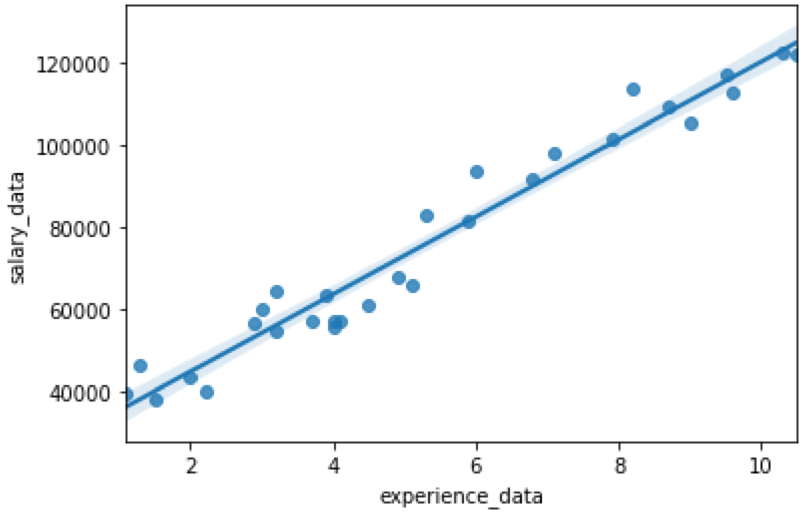
	experience_data	salary_data
experience_data	1.000000	0.978242
salary_data	0.978242	1.000000

In [27]:

```
sns.regplot(x='experience_data',y='salary_data', data=salary_Data,)
```

Out[27]:

```
<AxesSubplot:xlabel='experience_data', ylabel='salary_data'>
```



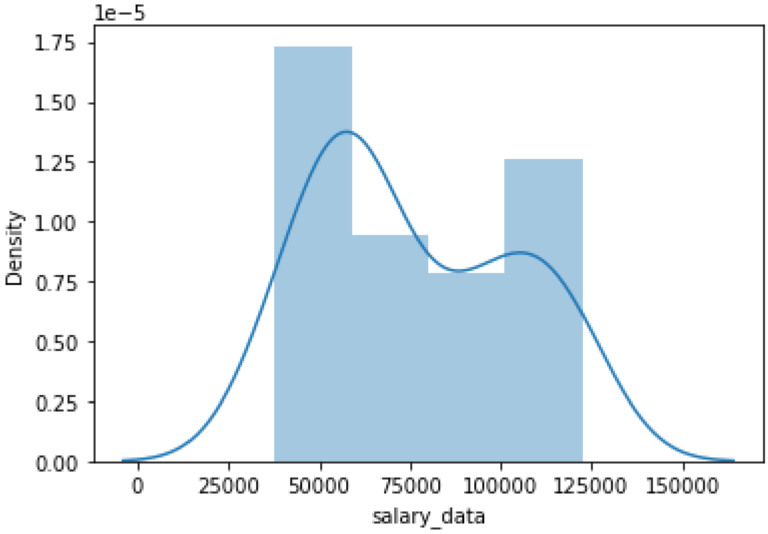
In [31]:

```
salary_Data.info()
sns.distplot(salary_Data['salary_data'])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   experience_data  30 non-null     float64
 1   salary_data      30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

Out[31]:

```
<AxesSubplot:xlabel='salary_data', ylabel='Density'>
```

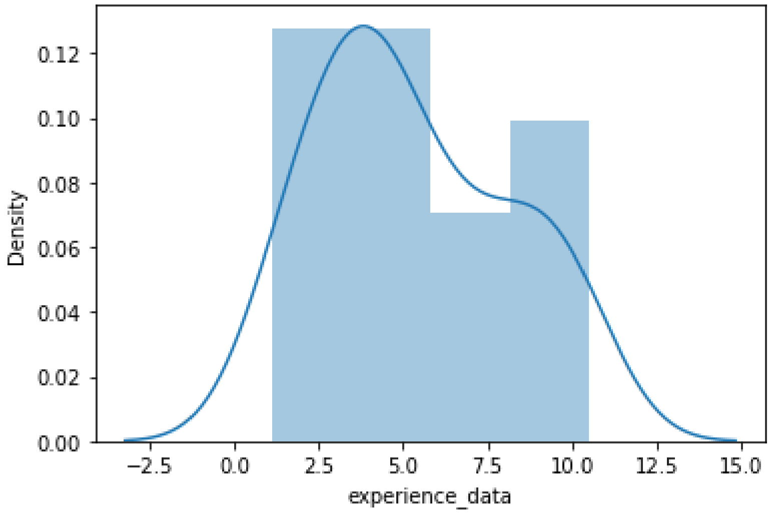


In [30]:

```
sns.distplot(salary_Data['experience_data'])
```

Out[30]:

```
<AxesSubplot:xlabel='experience_data', ylabel='Density'>
```



5. Model Building || Model Training

There are basically 2 libraries that support Leniar Regression algorithm

1. Statsmodels libraries

2. sklearn libraries

In [34]:

```
linear_model = smf.ols(formula = 'salary_data~experience_data',data = salary_Data).fit()
linear_model
```

Out[34]:

```
<statsmodels.regression.linear_model.RegressionResultsWrapper at 0x1b5394e5a
f0>
```

7. Model Testing

In [35]:

```
#Finding Coefficient parameters
linear_model.params
```

Out[35]:

```
Intercept          25792.200199
experience_data     9449.962321
dtype: float64
```

In [36]:

```
# Finding tvalues and pvalues
linear_model.tvalues, linear_model.pvalues
```

Out[36]:

```
(Intercept          11.346940
 experience_data    24.950094
 dtype: float64,
 Intercept          5.511950e-12
 experience_data    1.143068e-20
 dtype: float64)
```

In [37]:

```
# Finding Rsquared values
linear_model.rsquared, linear_model.rsquared_adj
```

Out[37]:

```
(0.9569566641435086, 0.9554194021486339)
```

8. Model prediction

Manual prediction for say sorting time 4

In [40]:

```
salary_Data = ( 25792.200199) + (9449.962321)*(4)
salary_Data
```

Out[40]:

```
63592.049483
```

9. Automatic prediction for say sorting time 4, 6

In [42]:

```
new_data = pd.Series([4,6])
new_data
```

Out[42]:

```
0    4
1    6
dtype: int64
```

In [43]:

```
data_pred = pd.DataFrame(new_data,columns = ['experience_data'])
data_pred
```

Out[43]:

	experience_data
0	4
1	6

In [45]:

```
linear_model.predict(data_pred)
```

Out[45]:

```
0    63592.049484
1    82491.974127
dtype: float64
```

In [ ]:

```
#Thanks Assignment Completed YearsExperience
#Question :- Predict YearsExperience using salary
#Manjunath Pujer 7th Nov 2021
```